# Handling numeric criteria in relaxed planning graphs

Oscar Sapena[1] and Eva Onaindía[2]

[1] Depto. de Ciencias de la Computación e Inteligencia Artificial,
Universidad de Alicante, Spain
osapena@dccia.ua.es
[2] Depto. de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, Spain
onandia@dsic.upv.es

**Abstract.** Nowadays, one of the main techniques used in heuristic planning is the generation of a relaxed planning graph, based on a *Graphplan*-like expansion. Planners like *FF* or *MIPS* use this type of graphs in order to compute distance-based heuristics during the planning process. This paper presents a new approach to extend the functionality of these graphs in order to manage numeric optimization criteria (problem metric), instead of only plan length optimization. This extension leads to more informed relaxed plans, without increasing significantly the computational cost. Planners that use the relaxed plans for further refinements can take advantage of this additional information to compute better quality plans.

## 1 Introduction

The problem of domain-independent planning is a very complex problem. In fact, the problem is *PSPACE-complete* even if some severe restrictions are applied [2]. Nowadays, one of the best techniques to deal with these complex problems is heuristic planning. Planners like *FF* [8], *LPG* [6], *VHPOP* [12] or *MIPS* [4], use different heuristic functions to guide the search, and all of them have demonstrated a very competitive performance.

The general principle for deriving heuristics is to formulate a simplified (or relaxed) version of the problem. Solving the relaxed problem is, in general, easier than solving the original problem. The solution of the relaxed problem is used as heuristic to estimate the distance to the goal. One of the most common relaxations is to ignore the negative effects (or delete list in *STRIPS* notation) of the actions. This idea was first proposed by *McDermott* [9] and, since then, a big number of planners have adopted this technique (i.e., *HSP* [1], *GRT* [10], *Sapa* [3], *FF*, etc.)

Nowadays, the planning community is working on extending the functionality of their planners to deal with more expressive problems. Many real-world problems have several characteristics that can hardly be expressed in pure propositional *STRIPS*: complex action conditions, numeric functions, durative actions,

uncertainty, etc. Most of these features can be modeled with the *PDDL 2.1* [5] language, but planners must be adapted to support the new extensions. One of the main contributions of *PDDL 2.1* is the possibility of specifying an optimization criterion for a planning problem. This criterion, called problem metric, consists of a numeric expression, which has to be maximized or minimized. Therefore, the user can ask the planner to optimize, for example, the fuel consumption in a transportation problem, rather than the commonly used criteria as the plan length or plan duration.

In this paper we propose a new technique to extend the relaxed planning graphs. This technique allows to deal with numeric optimization criteria. In spite of there are heuristic planners that can handle problem metrics (like *Metric-FF* [7] or *MIPS* [4]), their heuristic functions are still based on the plan length of the relaxed problem solution. This paper shows how this technique can be applied to improve the quality of the heuristic functions and, consequently, the quality of the final plans.

## 2 The relaxed planning graph

The relaxed planning graph (*RPG*) is a graph, based on a *Graphplan*-like expansion where delete effects are ignored. In this section we describe the traditional generation of a *RPG*, although taking into account the numeric part of *PDDL 2.1*. Firstly, we have to formalize some concepts. A (numeric) planning problem is defined as a tuple $\langle F, P, A, I, G, M\rangle$, where:

- $F$ is a set of numeric variables, called fluents.
- $P$ is a set of logical propositions.
- $A$ is the set of actions.
- $I$ is the initial state.
- $G$ is a set of goals.
- $M$ is the problem metric (numeric optimization criterion).

The numeric extensions of *PDDL 2.1* imply that, in addition to the propositions $P$, we have a set $F$ of numeric variables (fluents). A state $s$ is thus defined as a set of propositions $p(s)$ and a set of rational numbers $v(s)$ that represent the values for each fluent in that state:

$$s = \langle p(s), v(s)\rangle / p(s) \subset P \wedge v(s) = \{v_1, \ldots, v_n\} : value(f_i, s) = v_i, \forall f_i \in F$$

An *expression* is an arithmetic expression over $F$ and the rational numbers, using the operators +, -, * and /. The value of an expression $exp$ in a state $s$ is represented as $value(exp, s)$. A *numeric constraint* is a triple $\langle exp, comp, exp'\rangle$ where $exp$ and $exp'$ are expressions, and $comp \in \{<, \leq, =, \geq, >\}$ is a comparator. A *numeric effect* is a triple $\langle f_i, ass, exp\rangle$ where $f_i \in F$ is a fluent, $ass \in \{:=, +=, -=, *=, /=\}$ is an assignment operator, and $exp$ is an expression. The outcome of applying a numeric effect in a state $s$, written

$nresult(\langle f_i, ass, exp\rangle, s)$, is another state in which the value of fluent $f_i$ has been modified with the value of $exp$, using the assignment operator $ass$.

Actions in this numeric framework can have numeric preconditions and effects. Therefore, the preconditions of an action $a \in A$ can be propositional, $pprec(a) \subset P$, or numeric constraints, $nprec(a)$. Likewise, the effects of $a$ can be propositional, $add(a)$ and $del(a)$ for positive and negative effects respectively, or numeric effects, $neff(a)$. Regarding the problem goals, we restrict ourselves to propositional goals ($G \subset P$) for simplicity. This is the same simplification that the *Sapa* planner [3] does. However, the techniques presented in this paper can be easily translated to other approaches (like *Metric-FF* [7], which only ignores decreasing numeric effects). Finally, the problem metric ($M$) is an expression which value must be minimized. A maximization problem can be turned into a minimization problem just multiplying the metric expression by -1. Now, we can describe a relaxed planning problem, which ignores all delete effects of all actions.

**Definition 1** *Assuming a planning problem* $P_p = \langle F, P, A, I, G, M\rangle$*, the relaxation* $a^+$ *of an action* $a \in A$*,* $a = \langle pprec(a), nprec(a), add(a), del(a), neff(a)\rangle$*, is defined as:*

$$a^+ = \langle pprec(a), \emptyset, add(a), \emptyset, \emptyset\rangle$$

*The relaxed planning problem is* $P_p^+ = \langle F, P, A^+, I, G, M\rangle$*, where* $A^+ = \{a^+/a \in A\}$.

The first step to compute a solution for a relaxed planning problem is to build the *RPG*. The traditional *RPG* building algorithm generates proposition and action levels alternately. The first level ($P_0$) is a proposition level which contains all the propositions that are true in the starting state ($s_0$). Action levels contain all actions that are applicable in the previous level, and the following proposition levels are extended with the add effects of these actions. The expansion of the *RPG* finishes when a proposition level containing all top-level goals is reached, or when it is not possible to apply any new action. Figure 1 shows this process.

```
t = 0; P_0 = s_0                          // Initialization
while G ⊈ P_t do                          // New expansion stage
    A_t = {a ∈ A/pprec(a) ⊆ P_t}          // Action level
    P_{t+1} = P_t ⋃ add(a), ∀a ∈ A        // Proposition level
    if P_{t+1} = P_t then fail endif
    t = t + 1
endwhile
```

**Fig. 1.** Traditional *RPG* expansion.

## 3 Handling optimization criteria

The basic idea to take into account the optimization criterion in the *RPG* stage is to include some information about the actions cost according to the problem metric. Nevertheless, considering all possible different situations which can arise due to modifications in the fluent values is unfeasible. Therefore, our proposal consists in evaluating (an estimate of) the cost of the actions in the current state ($s_0$). This simplification is completely acceptable in problems where the costs and the consumption of resources are not very dependent on the state in which the actions are applied. Most of the planning problems fulfill this requirement. For example, the fuel consumption of a plane highly depends on static information like the flight distance. State-dependent information, like the number of passengers in a particular flight, only affects slightly.

The main difference with respect to the traditional *RPG* is that the levels of the graph do not represent time steps, but costs according to the problem metric ($M$). Thus, to compute the levels we have to estimate the cost of applying an action $a$:

$$\begin{array}{l} cost(a) = value(M, nresult(neff(a), s_0)) - value(M, s_0) + \varepsilon \\ \texttt{if } cost(a) < \varepsilon \texttt{ then } cost(a) = \varepsilon \texttt{ endif} \end{array} \quad (1)$$

The cost of an action $a$ is computed as the increase in the metric value caused by the application of $a$ in $s_0$ (we apply the numeric effects of $a$ to $s_0$, regardless of whether the preconditions of $a$ hold in $s_0$ or not). The small $\varepsilon$ included in the cost represents that every action, even those that do not affect the metric value, have a cost. This way, if there is no metric defined in the problem, our *RPG* is equivalent to the traditional one. Finally, we check the cost to be positive. We ignore the action effects which decrease the metric value, since it will cause an out of order expansion of the graph. Then, these actions are considered to have the minimum cost. The algorithm for the *RPG* expansion can be formalized as figure 2 shows.

The algorithm uses the list *prog_prop* (programmed propositions) in order to store the propositions which will be inserted in the graph. Each programmed proposition $p$ has an associated cost $cost(p)$. Initially, the *prog_prop* list only contains the propositions that hold in the current state $s_0$. These propositions have no cost since they are currently true. The rest of the propositions are not achieved yet and, therefore, have an infinite cost.

The graph expansion starts with the generation of the first propositional level. The propositional levels ($P_c$) are indexed through the cost of their propositions, so all propositions in a level will have the same cost. The level cost ($c$) is computed as the minimum cost of the programmed propositions, in order to build the graph from lower to higher cost values. The respective action level ($A_c$) contains the actions which preconditions have a cost $c$ or lower. The positive effects of these actions will be added to the *prog_prop* list only if they have not been achieved before with a lower cost. Let's suppose that $a$ is the action that produces a proposition $p$; the cost of $p$ ($cost(p)$) is computed as the addition of:

$prog\_prop = s_0$ // Initialization

$cost(p) = \begin{cases} 0 & \texttt{, if } p \in s_0 \\ \infty & \texttt{, otherwise} \end{cases}, \forall p \in P$

**while** $\exists g \in G / cost(g) = \infty$ **do** // New expansion stage

  **if** $prog\_prop = \emptyset$ **then fail endif**

  $c = min(cost(p)), \forall p \in prog\_prop$ // Level cost

  $P_c = \{p / p \in prog\_prop \wedge cost(p) = c\}$ // Proposition level

  $prog\_prop = prog\_prop - P_c$

  $A_c = \{a \in A / cost(p) \leq c, \forall p \in pprec(a)\}$ // Action level

  **for all** $a \in A_c$ **do** // Programming action effects

    $cost\_reach(a) = \sum cost(p), \forall p \in pprec(a)$

    **for all** $p \in add(a) \wedge (cost\_reach(a) + cost(a) < cost(p))$ **do**

      $prog\_prop = prog\_prop \cup \{p\}$

      $cost(p) = cost\_reach(a) + cost(a)$

    **endfor**

  **endfor**

**endwhile**

**Fig. 2.** Proposed *RPG* for problem metric optimization.

- The cost of achieving $a$ ($cost\_reach(a)$): this cost is computed as the sum of the $a$ preconditions costs.
- The cost of applying $a$ ($cost(a)$), defined in (1).

The *RPG* expansion finishes when all top-level goals are achieved, or when the *prog_prop* list becomes empty. If the *prog_prop* list is empty, then no new action can be applied and, therefore, the goals cannot be achieved.

This algorithm can be used to improve the heuristic information extracted from the relaxed plans. Section 3.1 shows an example to compare the traditional *RPG* with our proposal. Moreover, the computational complexity of the algorithm is polynomial since the traditional one is proved to be polynomial [8], and the additional calculations (action and proposition costs) can be done in polynomial time.

### 3.1 Example of the *RPG* expansion

We will illustrate the *RPG* expansion through a *Logistics*-like example problem (see figure 3). In this problem, there are three cities ($C_1$, $C_2$ and $C_3$), one truck ($T$) and one package ($P$). The truck and the package are initially located in $C_1$ and $C_2$ respectively. The distance between the cities is shown in figure 3 through the labels in the roads. The goal is to carry the package to city $C_3$, minimizing the driven distance (the problem metric is: $minimize(driven)$). Table 1 summarizes the domain description.

The *RPG* for this problem is shown in table 2. This *RPG* has more levels than the traditional one, but the number of actions and propositions per level is lower. The first action level ($A_0$) contains the actions that are directly executable: drive from $C_1$ to $C_2$ and $C_3$. The cost of having truck $T$ in $C_3$ is 5 (since this is the

**Table 1.** Domain description of the example problem.

| operator | param. | pprec | add | del | neff |
|---|---|---|---|---|---|
| Load | ?c | at T ?c ∧ at P ?c | in P T | at P ?c | ∅ |
| Unload | ?c | at T ?c ∧ in P T | at P ?c | in P T | ∅ |
| Drive | ?c1 ?c2 | at T ?c1 | at T ?c2 | at T ?c1 | driven+=distance ?c1 ?c2 |

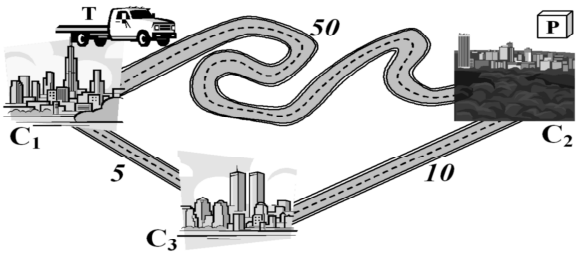


**Fig. 3.** Initial state in the logistics example problem.

distance between $C_1$ and $C_3$), so the action effects ($at\ T\ C_3$) will be programmed with the cost $5 + \varepsilon$. Actions which do not affect the metric, like the load and unload operations, have a cost of $\varepsilon$. The expansion finishes when the goal ($at\ P\ C_3$) is achieved.

**Table 2.** Relaxed planning graph for the example problem.

| $P_0$ | $A_0$ | $P_{5+\varepsilon}$ | $A_{5+\varepsilon}$ | $P_{15+2\varepsilon}$ | $A_{15+2\varepsilon}$ |
|---|---|---|---|---|---|
| $at\ T\ C_1$ | $Drive\ C_1\ C_2$ | $at\ T\ C_3$ | $Drive\ C_3\ C_1$ | $at\ T\ C_2$ | $Drive\ C_2\ C_1$ |
| $at\ P\ C_2$ | $Drive\ C_1\ C_3$ | | $Drive\ C_3\ C_2$ | | $Drive\ C_2\ C_3$ |
| | | | | | $Load\ C_2$ |
| $P_{15+3\varepsilon}$ | $A_{15+3\varepsilon}$ | $P_{15+4\varepsilon}$ | $A_{15+4\varepsilon}$ | $P_{20+5\varepsilon}$ | |
| $in\ P\ T$ | $Unload\ C_1$ | $at\ P\ C_1$ | $Load\ C_1$ | $\underline{at\ P\ C_3}$ | |
| | $Unload\ C_2$ | | | | |
| | $Unload\ C_3$ | | | | |

The relaxed plan obtained using our *RPG* is the following:

$P1 = Drive\ C_1\ C_3 \longrightarrow Drive\ C_3\ C_2 \longrightarrow Load\ C_2 \longrightarrow Unload\ C_3$

The following plan has been computed by means of the *FF's* relaxed plan extraction algorithm [7]:

$P2 = \{Drive\ C_1\ C_2, Drive\ C_1\ C_3\} \longrightarrow Load\ C_2 \longrightarrow Unload\ C_3$

The benefits of our proposal can be easily observed just comparing both relaxed plans. It can be observed that $P1$ is almost executable (it only needs one

action to go from $C_2$ to $C_3$). Plan $P2$, however, has hard conflicts since actions *Drive* $C_1$ $C_2$ and *Drive* $C_1$ $C_3$ are mutually exclusive. Moreover, action *Drive* $C_1$ $C_2$ should not be included in the plan because of its high cost. Obviously, if the planner only takes into account the relaxed plan length to compute its heuristics, this technique does not bring many advantages. On the contrary, planners that use the information provided by relaxed plan will find a valuable help building the final plan.

## 4 Results

The *RPG* expansion for handling numeric criteria has been implemented in the *SimPlanner v3* planner [11]. *SimPlanner v3* is a heuristic planner that can take advantage of the presented improvements on the relaxed planning graphs. In other heuristic planners that only use the relaxed plan length, like *FF* or *HSP*, the extra valuable information from the *RPG* is not fully exploited.

Our proposed expansion and the traditional one are compared in this section. Both techniques are implemented in the same planner (*SimPlanner v3*). This way, the results are not influenced by other characteristics of the planner. For this reason, we have not included comparisons between *SimPlanner v3* and other heuristic planners, since the results would not provide reliable information about the performance of our proposal.

The domains used in this comparison are numeric domains introduced in the third international planning competition (IPC'02). The domains used in the competition are described at

`http://planning.cis.strath.ac.uk/competition.`

Table 3 shows the results obtained in this comparison for the problems in *DriverLog*, *ZenoTravel* and *Depots* numeric domains. The *DriverLog* is a variation of *Logistics* where trucks need drivers. Drivers can move along different road links than trucks. The optimization criterion in *DriverLog* is to minimize an instance-specific linear combination of total time, driven distance and walked distance. *ZenoTravel* is a transportation domain, where objects must be transported via aeroplanes. The optimization criterion is to minimize an instance-specific linear combination of total time and fuel consumption. The *Depots* domain is a combination of *Logistics* and *Blockworld* domains, where some blocks must be transported with trucks between depots and arranged in a certain order. The optimization criterion is to minimize the overall fuel consumption.

Results on table 3 show the plan quality of the solutions, according to the problem metric defined, and the running times. These results show that our proposal improves the plan quality in most of the problems, and solves more problems than using the traditional approach. On average, the computed plans are 1.87, 1.25 and 1.23 times better in the *DriverLog*, *ZenoTravel* and *Depots* domains respectively. However, due to the heuristic nature of *SimPlanner v3*, there are a few problems where the traditional approach obtains better plans.

Regarding the running times, table 3 shows that our proposal takes more time than the traditional one. This is mainly due to three factors:

**Table 3.** Comparison between the results obtained with the proposed expansion and with the traditional one (in the form *proposed/traditional*) for the numeric *DriverLog*, *ZenoTravel* and *Depots* domains. Quality depends on the problem metric (greater numbers stand for more costly plans), and time is measured in seconds.

| | DriverLog | | ZenoTravel | | Depots | |
|---|---|---|---|---|---|---|
| Prob. | Quality | Time | Quality | Time | Quality | Time |
| 1 | 777/777 | 0.01/0.01 | 13564/13564 | 0.01/0.01 | 32/42 | 0.01/0.01 |
| 2 | 999/1625 | 0.08/0.13 | 6786/6786 | 0.01/0.01 | 43/53 | 0.04/0,03 |
| 3 | 1406/1406 | 0.03/0.03 | 6758/6758 | 0.01/0.01 | 29/29 | 0.22/0.15 |
| 4 | 1119/986 | 0.05/0.05 | 27000/27000 | 0.01/0.01 | 64/50 | 0.54/0.44 |
| 5 | 1056/1270 | 0.05/0.05 | 3978/3978 | 0.02/0.02 | 80/259 | 0.92/2.54 |
| 6 | 2095/2466 | 0.03/0.02 | 25097/25097 | 0.03/0.03 | 313/313 | 205.2/195.5 |
| 7 | 1876/1476 | 0.07/0.04 | 11198/11198 | 0.02/0.02 | 37/57 | 0.1/0.1 |
| 8 | 3418/3472 | 0.17/0.09 | 29677/55480 | 0.04/0.04 | 43/43 | 0.43/0.43 |
| 9 | 4091/9144 | 1.43/1.53 | 13275/12644 | 0.14/0.06 | 231/409 | 88.47/79.92 |
| 10 | 241/3211 | 0.07/0.26 | 177368/175218 | 0.36/0.2 | 27/27 | 0.27/0.27 |
| 11 | 753/738 | 0.18/0.07 | 25505/65093 | 0.15/0.05 | 229/196 | 8.35/13.05 |
| 12 | 6713/5346 | 5.86/0.7 | 52206/38547 | 0.2/0.08 | 299/389 | 34.09/27.54 |
| 13 | 2886/3556 | 0.83/0.76 | 112468/95657 | 0.55/0.22 | 27/27 | 1.93/2 |
| 14 | 11153/- | 2.22/- | 430417/183591 | 3.51/0.27 | 43/43 | 2.4/2.42 |
| 15 | 3561/3464 | 2.79/0.59 | 59835/205701 | 1.92/1.65 | 237/214 | 28.21/34.35 |
| 16 | 16829/39771 | 110.6/24 | 64119/87530 | 4.14/1.48 | 31/32 | 0.29/0.27 |
| 17 | 20655/93031 | 60/27.7 | 190845/384645 | 36.94/12.36 | 29/31 | 0.53/0.53 |
| 18 | 70917/82266 | 283.4/91.3 | 67610/71944 | 20.42/7.26 | 108/127 | 37.4/36.73 |
| 19 | -/- | -/- | 235378/257104 | 44.46/15.2 | 48/48 | 4.65/4.7 |
| 20 | 11555/27884 | 462/70.7 | 111671/355711 | 26.91/20.21 | 217/- | 28.46/- |
| | 1.87 better | 4.26 slower | 1.25 better | 2.36 slower | 1.23 better | 1.03 slower |

- **The application of formula (1) to estimate the actions costs.** However, in these domains this computation has only a slight effect on the running time since the costs are static, that is, they do not depend on the state. And, for the same reason, the estimated costs for these domains are only computed once in the planning process.
- **The number of graph levels.** This number is from 10 to 70 times greater in our proposal, although the number of actions per level is lower.
- **The number of actions in the graph.** This number is often greater than in the traditional approach due to the expansion of many unuseful low-cost actions in first place. In the *DriverLog* domain, for example, a lot of unnecessary *walk* actions are inserted in the graph. This is the main reason for the greater running times in most of the problems.

These three factors slightly increase the time consumed in the *RPG* creation. However, the overall time increment is more significant as *SimPlanner v3* has to build many *RPGs* when solving a problem. For example, the number of created *RPGs* is greater than $10^5$ for some problems.

## 5 Conclusions and future work

In this paper, we have presented an extension to the traditional relaxed planning graph generation. A relaxed planning graph is based on a GraphPlan-like expansion, where delete effects are ignored. The use of these graphs is widely used in many heuristic planners.

The proposed extension allows to take into account the optimization criterion (or metric) of the problem. During the graph expansion, an estimate of the cost of the actions is computed according to the problem metric. This estimate is used to expand the less costly actions in first place. Practical results show that our proposal, in general, obtains better solutions than the traditional approach.

The relaxed planning graphs are the starting point for the heuristic estimators of many planners. Therefore, all the improvements on these graphs can help the planners increase their performance. *Metric-FF* [7], for example, proposes an extension to deal with the numeric effects of the actions. The *Metric-FF* expansion is completely compatible with our proposed expansion and, therefore, both techniques could be implemented in the same planner. However, there are several features that have not been addressed in the relaxed planning graph framework yet. Handling probabilities and sensing actions, for example, could allow the planners to face problems with uncertainty more efficiently.

## Acknowledgements

This work has been partially funded by projects *SAMAP TIC2002-04146-C05-04*, *MCyT DPI2001-2094-C03-03* and *UPV 20020681*.

## References

1. B. Bonet and H. Geffner, 'Planning as heuristic search', *Artificial Intelligence Journal*, **129**, 5–33, (2001).
2. T. Bylander, 'The computational complexity of propositional STRIPS planning', *Artificial Intelligence*, **69**, 165–204, (1994).
3. M.B. Do and S. Kambhampati, 'Sapa: A multi-objective metric temporal planner', *Journal of Artificial Intelligence Research*, **20**, 155–194, (2003).
4. S. Edelkamp, 'Taming numbers and durations in the model checking integrated planning system', *Journal of Artificial Intelligence Research*, **20**, 195–238, (2003).
5. M. Fox and L. Derek, 'PDDL2.1: An extension to PDDL for expressing temporal planning domains', *Journal of Artificial Intelligence Research*, **20**, 61–124, (2003).
6. A. Gerevini, A. Saetti, and I. Serina, 'Planning through stochastic local search and temporal action graphs in LPG', *Journal of Artificial Intelligence Research*, **20**, 239–290, (2003).
7. J. Hoffmann, 'The Metric-FF planning system: Translating 'ignoring delete lists' to numeric state variables', *Journal of Artificial Intelligence Research*, **20**, 291–341, (2003).
8. J. Hoffmann and B. Nebel, 'The FF planning system: Fast planning generation through heuristic search', *Journal of Artificial Intelligence Research*, **14**, 253–302, (2001).

9. D. McDermott, 'A heuristic estimator for means-ends analysis in planning', *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, **3**, 142–149, (1996).
10. I. Refanidis and I. Vlahavas, 'The GRT planning system: Backward heuristic construction in forward state-space planning', *Journal of Artificial Intelligence Research*, **14**, 115–161, (2001).
11. O. Sapena, E. Onaindia, M. Mellado, C. Correcher, and V. Vendrell, 'Reactive planning simulation in dynamic environments with VirtualRobot', *Innovations in Applied Artificial Intelligence, LNCS*, **3029**, 699–707, (2004).
12. H.L.S. Younes and R.G. Simmons, 'VHPOP: Versatile heuristic partial order planner', *Journal of Artificial Intelligence Research*, **20**, 405–430, (2003).